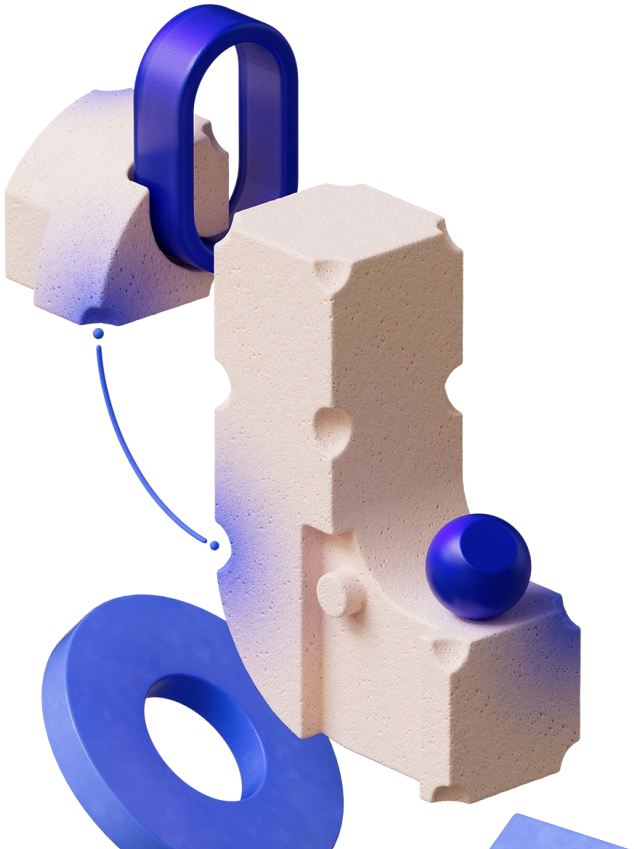
Google DeepMind



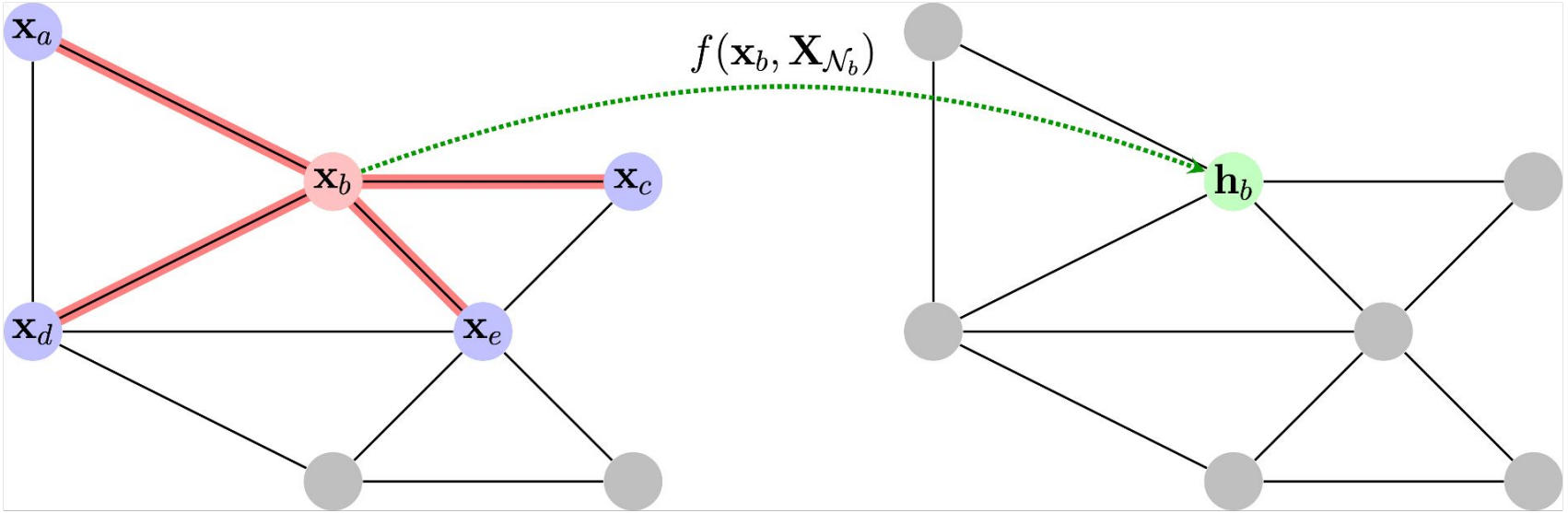
Decoupling the *computational* graph and the *input* graph

The **most** important *unsolved* problem in graph representation learning

Petar Veličković
Staff Research Scientist, Google DeepMind
Affiliated Lecturer, University of Cambridge

Explainability and Applicability of Graph Neural Networks
7 September 2023

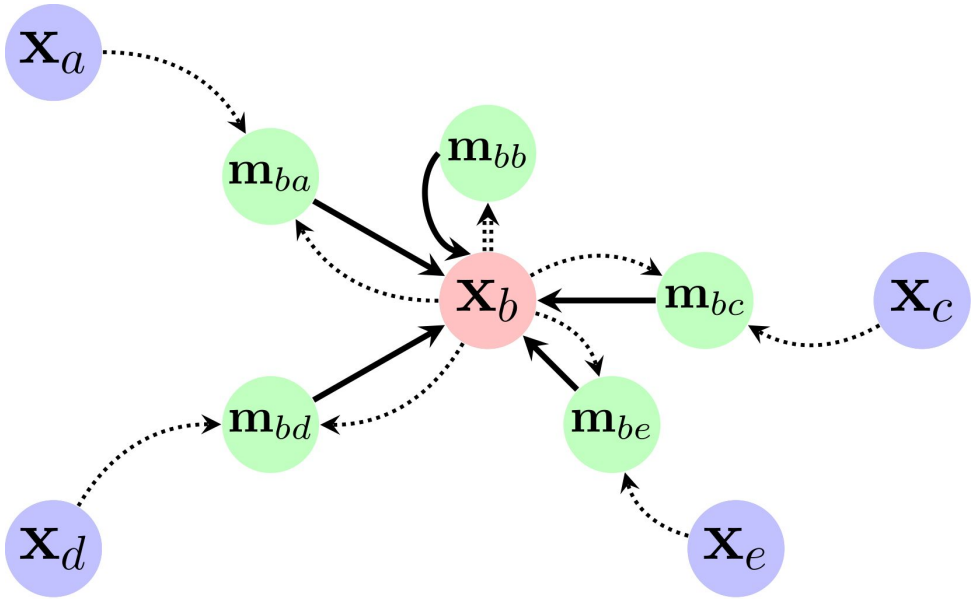
Our key player: the *graph* neural network (**GNN**)



$$\mathbf{X}_{\mathcal{N}_b} = \{\{\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_d, \mathbf{x}_e\}\}$$

The local function, *f*, needs to be **permutation invariant** to neighbourhood features $\mathbf{X}_{\text{Ni}}$

Zooming into GNNs: the *message-passing* paradigm

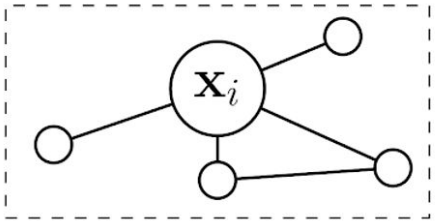


Message-passing

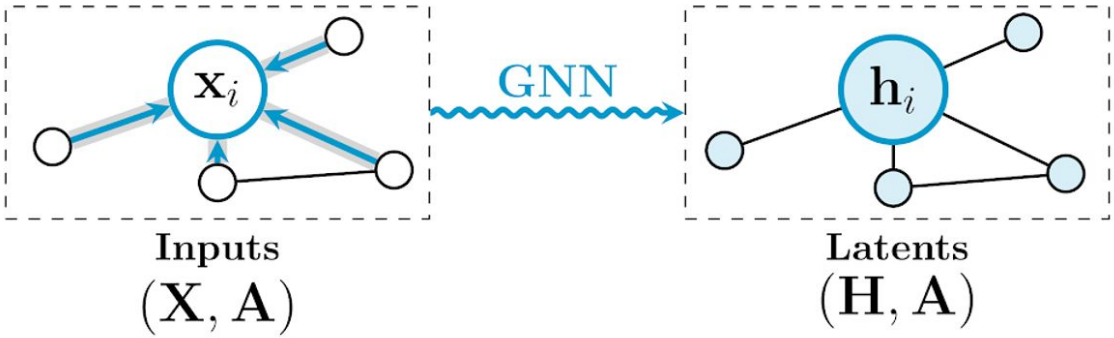
$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

The aggregation function, $\oplus$, needs to be **permutation invariant** (e.g. sum, max, average)

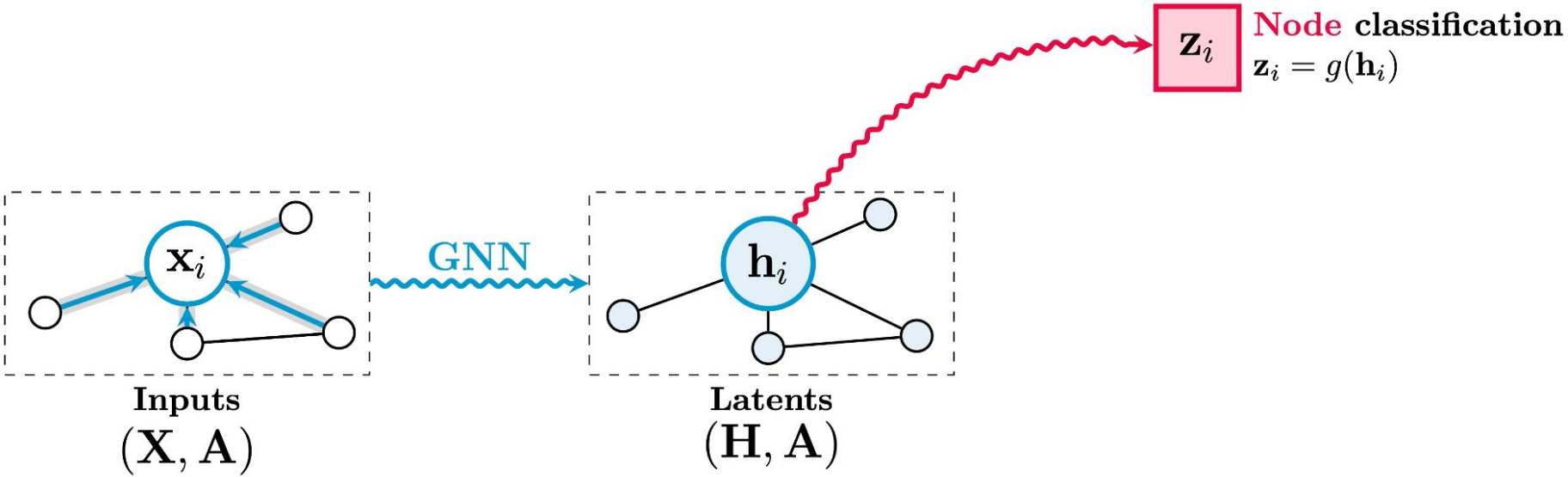
Using GNNs to solve *everything else*



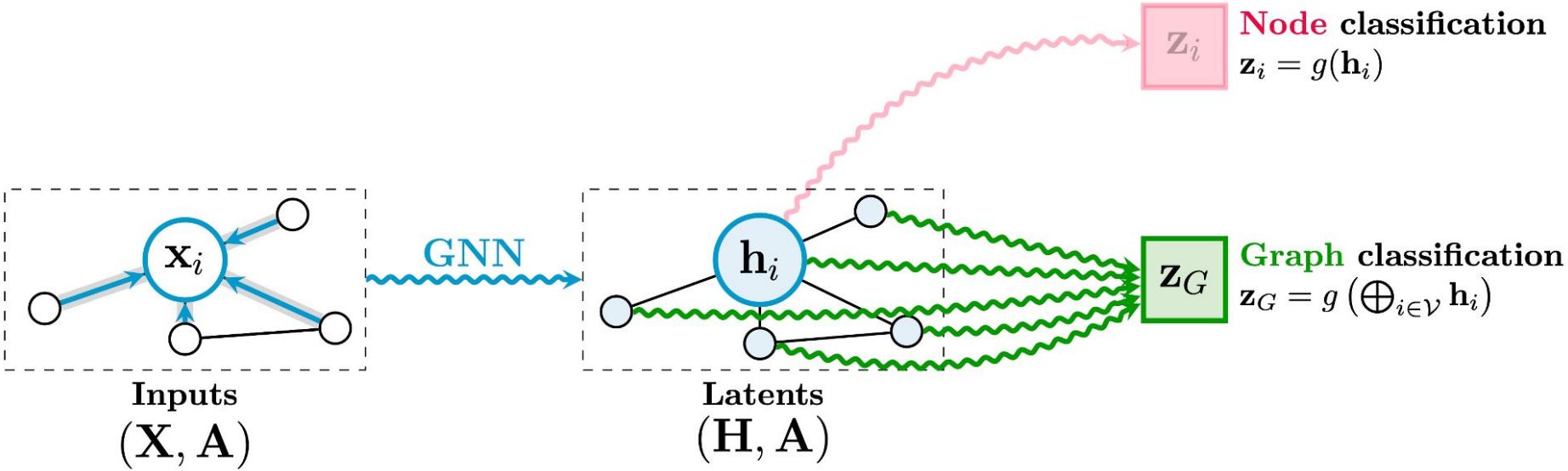
Using GNNs to solve *everything else*



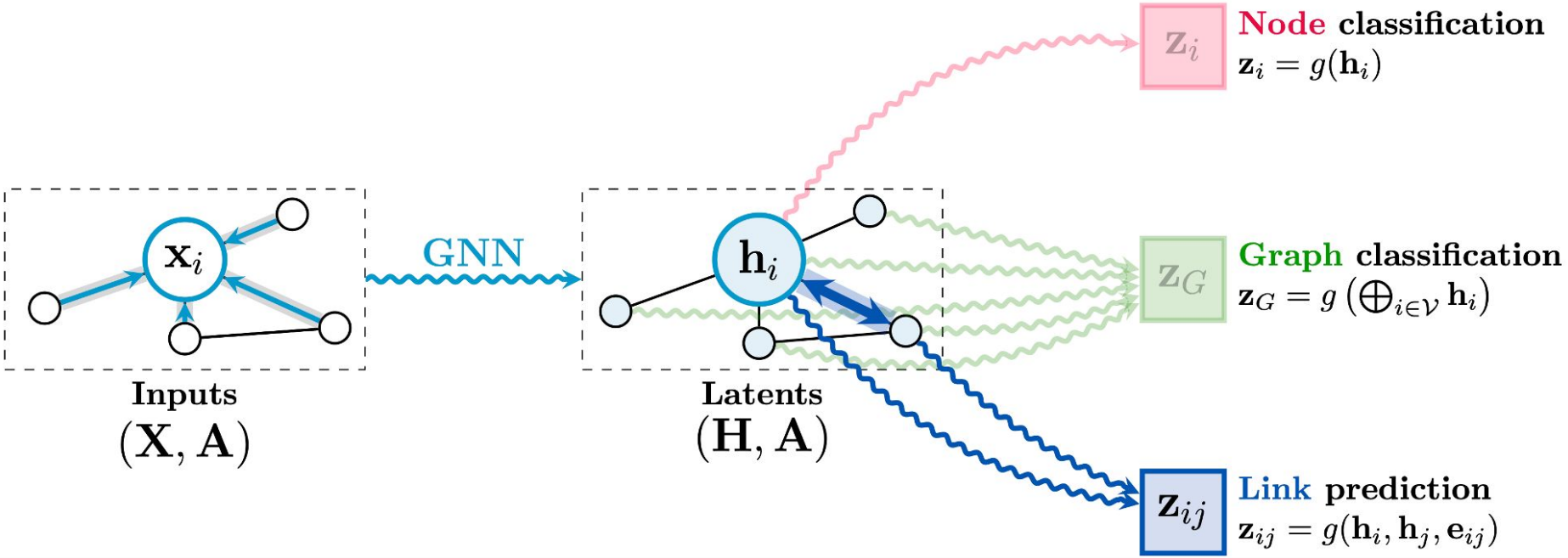
Using GNNs to solve *everything else*



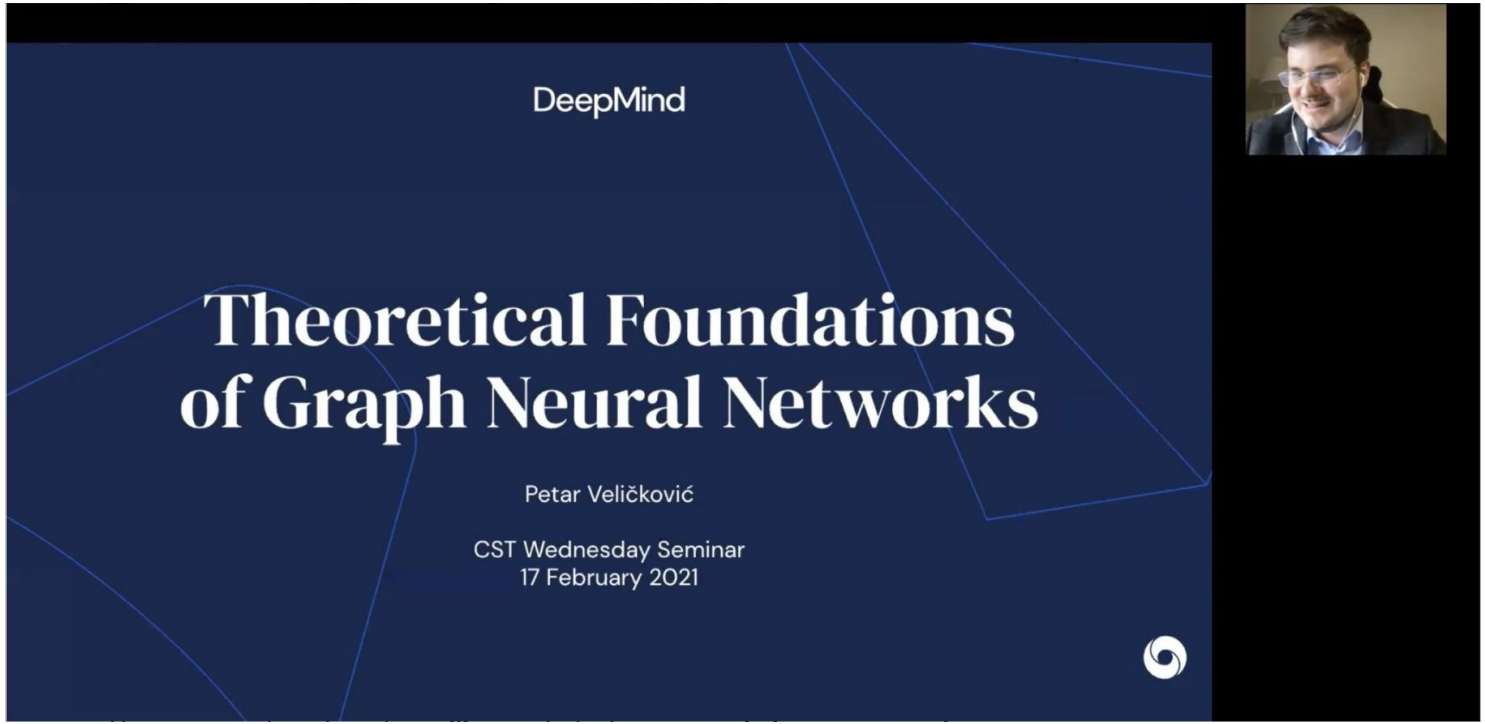
Using GNNs to solve *everything else*



Using GNNs to solve *everything else*



For more information on GNNs...



This is as far as I'm able to talk about GNNs here... but my prior GNN talks are publicly available :)

Towards *graph rewiring*

- The GNN equation assumes something (seemingly) very innocent: that the *"ground truth" graph is given to us!*

- However, this claim is far from innocent, as we will unpack now
  - It will lead us into a dynamic area known as **graph rewiring**
  - A key emerging area of graph representation learning (and an area of high interest to science)

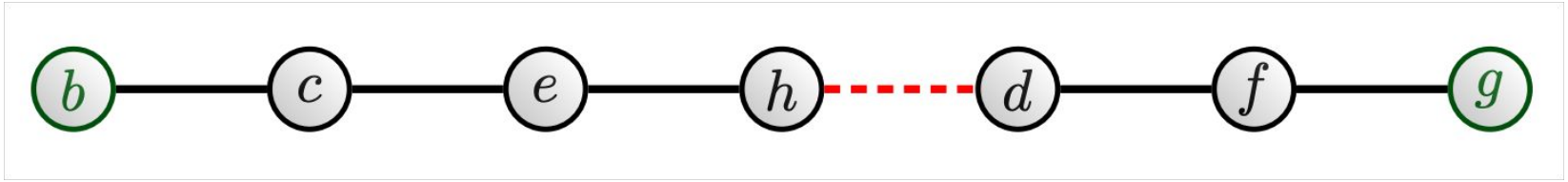
- Further, studying this area will reveal surprising connections between state-of-the-art models and graph neural networks *(Spoiler alert: almost everything is a special case of a GNN :))*

The importance of a *"bespoke"* graph

In practice, the given graph is often *suboptimal* for the task, *even if it is* ***completely*** *correct*

Imagine you want to repeatedly query whether pairs of nodes are *connected* in a graph

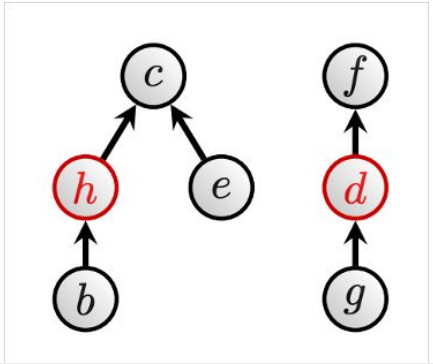
Naïvely running a GNN over the *exact* edges of the graph requires **linear time** in the worst case!

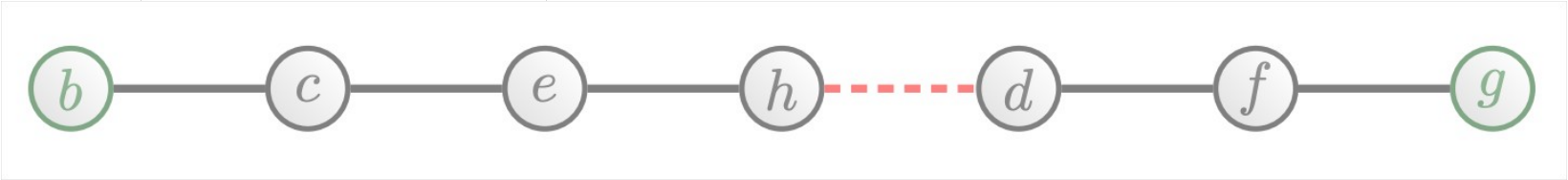


Imagine we **query** connectivity of *(b, g)* after **adding** edge *(h, d)*...

The importance of a *“bespoke”* graph

Maintaining the right set of edges can make a difference between **linear-time** and (amortised) **near-constant-time** complexity!

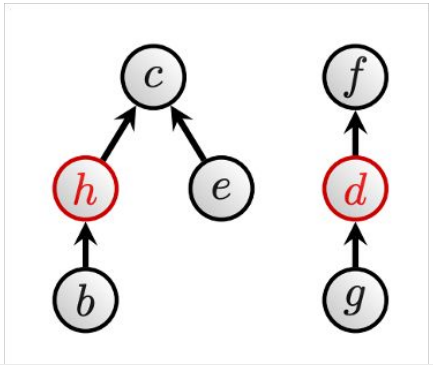


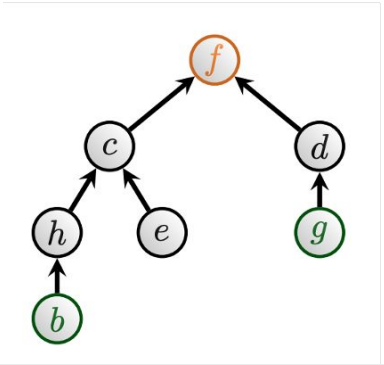


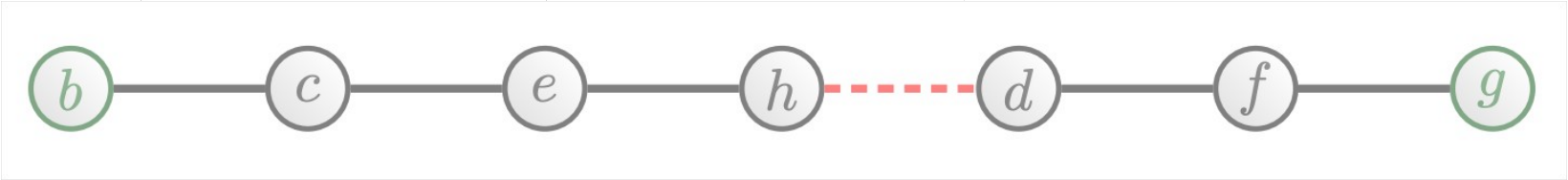
Here, we use a disjoint-set union data structure, which stores a *forest* that identifies each *connected component* with one *tree* (and its **root**).

The importance of a *“bespoke”* graph

Maintaining the right set of edges can make a difference between **linear-time** and (amortised) **near-constant-time** complexity!



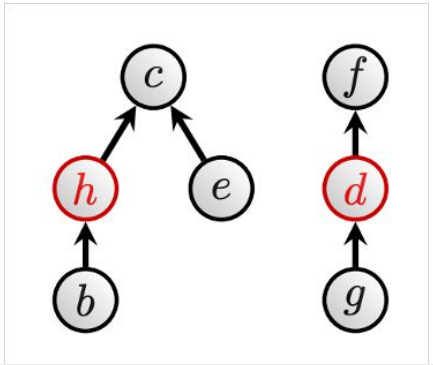


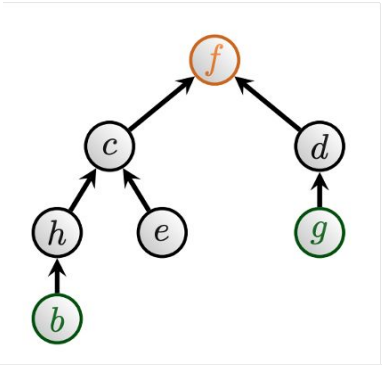


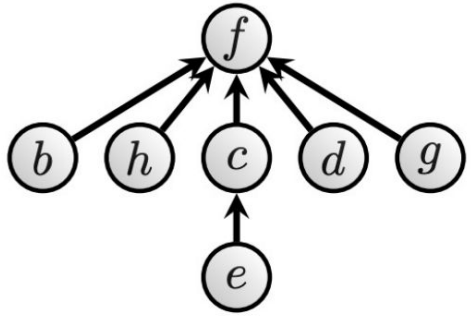
After **adding** a new edge *(h, d)*, the respective trees are *merged*

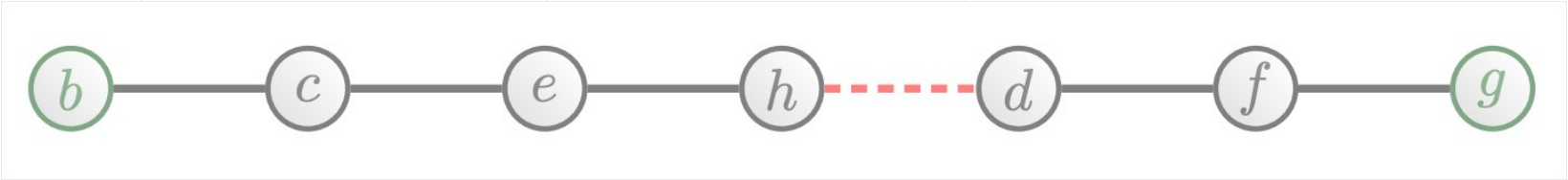
The importance of a *“bespoke”* graph

Maintaining the right set of edges can make a difference between **linear-time** and (amortised) **near-constant-time** complexity!







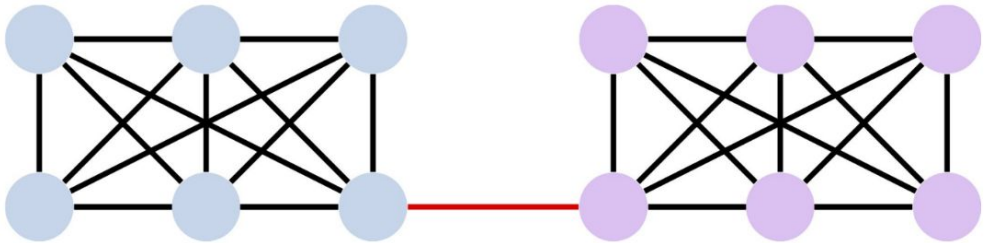


After **querying** *(b, g)*, the tree gets *compressed* for fast future lookups

Bottlenecks and oversquashing

- Additionally, certain graph structures can induce *pathological* effects in GNN propagation, no matter how well they are trained!

- For example, issues can often arise around **bottlenecks**:



- Bottlenecks cause the ***oversquashing*** effect (Alon and Yahav, ICLR'21)

- To meaningfully allow *all* pairs of nodes to communicate, the message size over the bottleneck needs to grow *exponentially* with GNN depth

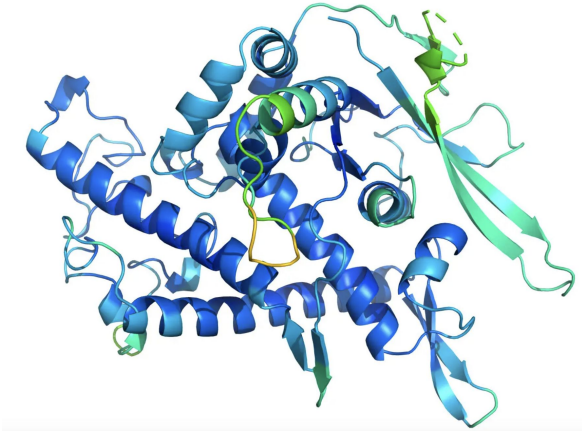
In summary...

- Even when a graph is **perfectly correct**, it might not be *optimal* for the task at hand
  - In the path-querying example, both graphs “fully correctly” encode the information
  - One induces O(*n*) GNN layers; the other induces near-O(1) layers

In summary...

- Even when a graph is **perfectly correct**, it might not be *optimal* for the task at hand
  - In the path-querying example, both graphs “fully correctly” encode the information
  - One induces O(*n*) GNN layers; the other induces near-O(1) layers

- In the **real world**, “perfect graphs” are **incredibly rare**
  - Take the standard example: a molecule!
  - It’s an excellent representation for *chemistry*, but only an approximation for *physics*
  - In practice, *all* pairs of atoms interact, not just ones connected by a bond!



In summary...

- Even when a graph is **perfectly correct**, it might not be *optimal* for the task at hand
  - In the path-querying example, both graphs “fully correctly” encode the information
  - One induces O(*n*) GNN layers; the other induces near-O(1) layers

- In the **real world**, “perfect graphs” are **incredibly rare**
  - Take the standard example: a molecule!
  - It’s an excellent representation for *chemistry*, but only an approximation for *physics*
  - In practice, *all* pairs of atoms interact, not just ones connected by a bond!

- Some graphs are just **pathologically bad**
  - No matter what GNN we can feasibly train over them...
  - Certain kinds of communication will be *practically impossible*!

    *(until quite recently, we did **not** know exactly **when** this happens...)*

In summary...

- Even when a graph is **perfectly correct**, it might not be *optimal* for the task at hand
  - In the path-querying example, both graphs “fully correctly” encode the information
  - One induces O(*n*) GNN layers; the other induces near-O(1) layers

- In the **real world**, “perfect graphs” are **incredibly rare**
  - Take the standard example: a molecule!
  - It’s an excellent representation for *chemistry*, but only an approximation for *physics*
  - In practice, *all* pairs of atoms interact, not just ones connected by a bond!

- Some graphs are just **pathologically bad**
  - No matter what GNN we can feasibly train over them...
  - Certain kinds of communication will be *practically impossible*!

    *(until quite recently, we did **not** know exactly **when** this happens...)*

- Choosing a computational graph may give us the **final** building block to *“all of discrete DL”*

One equation for ***all*** of (discrete) deep learning?

Let $\text{GNN}_\theta(\mathcal{G}, \mathbf{X})$ be *any* neural network with parameters θ applied over the graph $\mathcal{G}$ with (node, edge…) features **X**

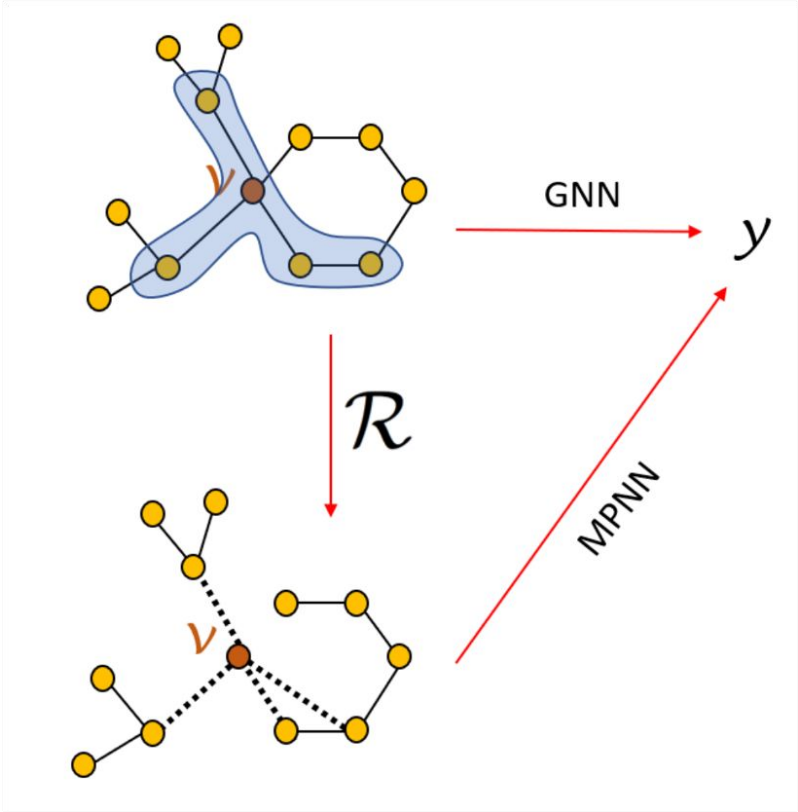
Then:

$$\text{GNN}_\theta(\mathcal{G}, \mathbf{X}) = \text{MPNN}_{\theta'}(\mathcal{R}(\mathcal{G}), \mathcal{R}(\mathbf{X}))$$

where MPNN is a message-passing GNN over a one-hop neighbourhood and $\mathcal{R}$ is *rewiring*

(NB. $\mathcal{R}$ may introduce new nodes/edges in $\mathcal{G}$ and appropriately initialise their features in **X**!)

See: Francesco Di Giovanni's keynote @ NeurIPS'22 GLFrontiers



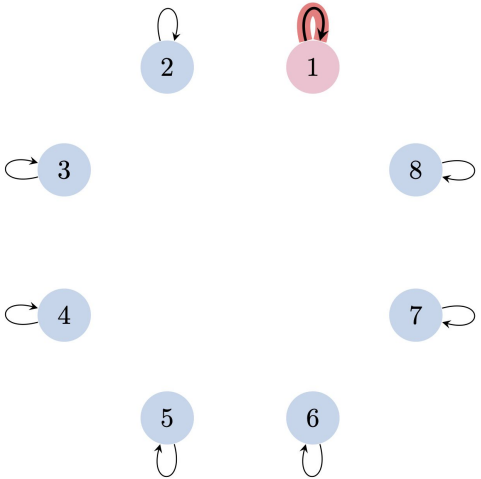
So... how do we *rewire* a graph?

So... how do we *rewire* a graph?

Assume ***no*** edges

So... how do we *rewire* a graph?

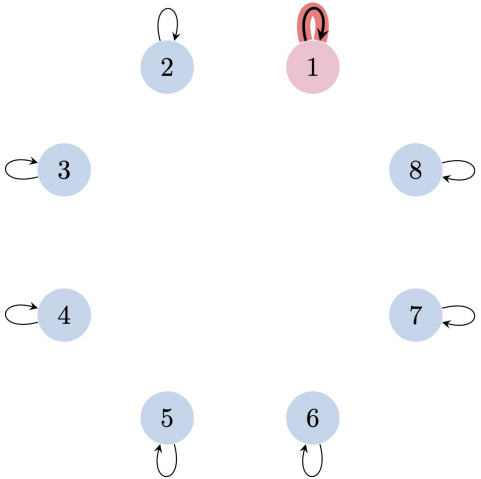
Assume ***no*** edges



$$\mathcal{N}_i = \{i\}$$

So... how do we *rewire* a graph?

Assume ***no*** edges



Under all GNN flavours, this boils down to:

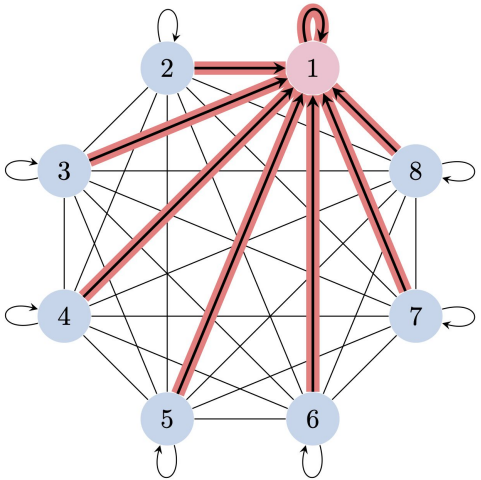
$$\mathbf{h}_i = \psi(\mathbf{x}_i) \qquad \mathcal{N}_i = \{i\}$$

Deep Sets (Zaheer *et al.*, NeurIPS'17)

So... how do we *rewire* a graph?

Assume ***no*** edges ⟶ Deep Sets

Assume ***all*** edges



If we use attentional GNNs:

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{V}} a(\mathbf{x}_i, \mathbf{x}_j)\psi(\mathbf{x}_j)\right) \qquad \mathcal{N}_i = \mathcal{V}$$

*(Does this look **familiar**?)*

A note on Transformers

Transformers **are** Graph Neural Networks!

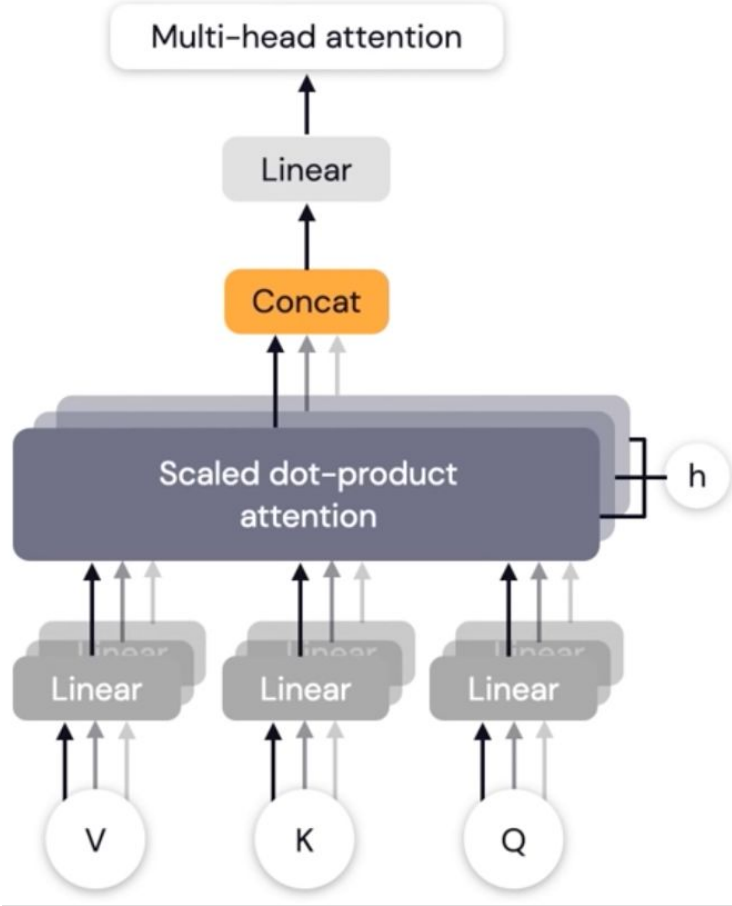
- Fully-connected graph
- Attentional flavour

The sequential structural information is injected through the **positional embeddings**. But the model is under no **requirement** to use this information!

Attention can be seen as inferring **soft adjacency**.

“GNNs that are winning the hardware lottery”

See Joshi (The Gradient; 2020).



So... how do we *rewire* a graph?

Assume ***no*** edges → Deep Sets

Assume ***all*** edges → Transformers

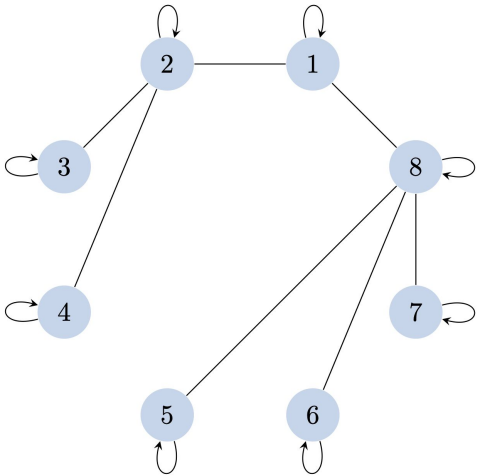
Nonparametric rewiring

So... how do we *rewire* a graph?

Assume ***no*** edges ⟶ Deep Sets

Assume ***all*** edges ⟶ Transformers

Nonparametric rewiring

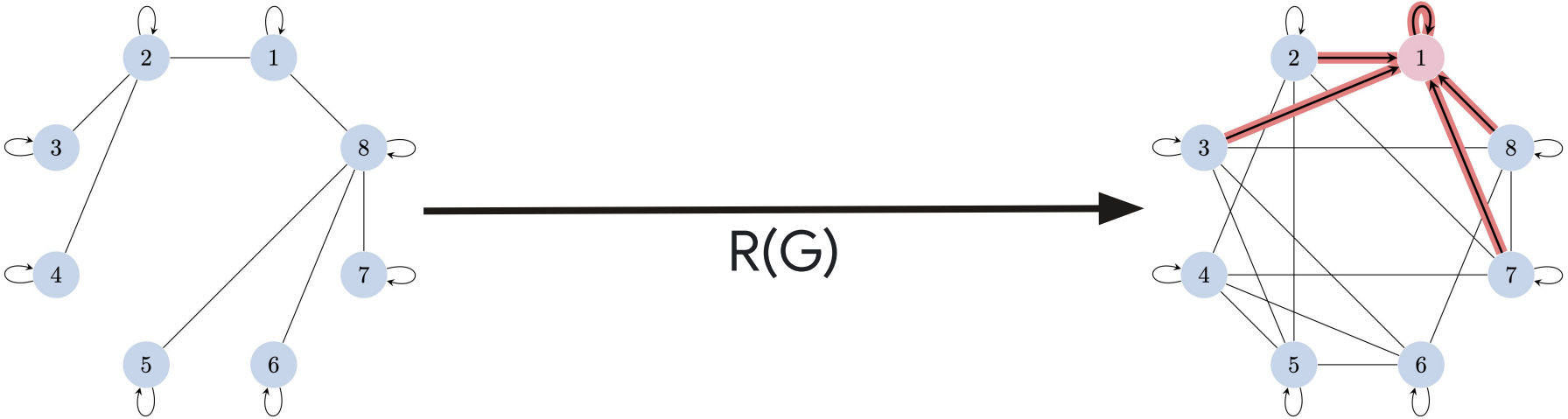


So... how do we *rewire* a graph?

Assume ***no*** edges ⟶ Deep Sets

Assume ***all*** edges ⟶ Transformers

Nonparametric rewiring



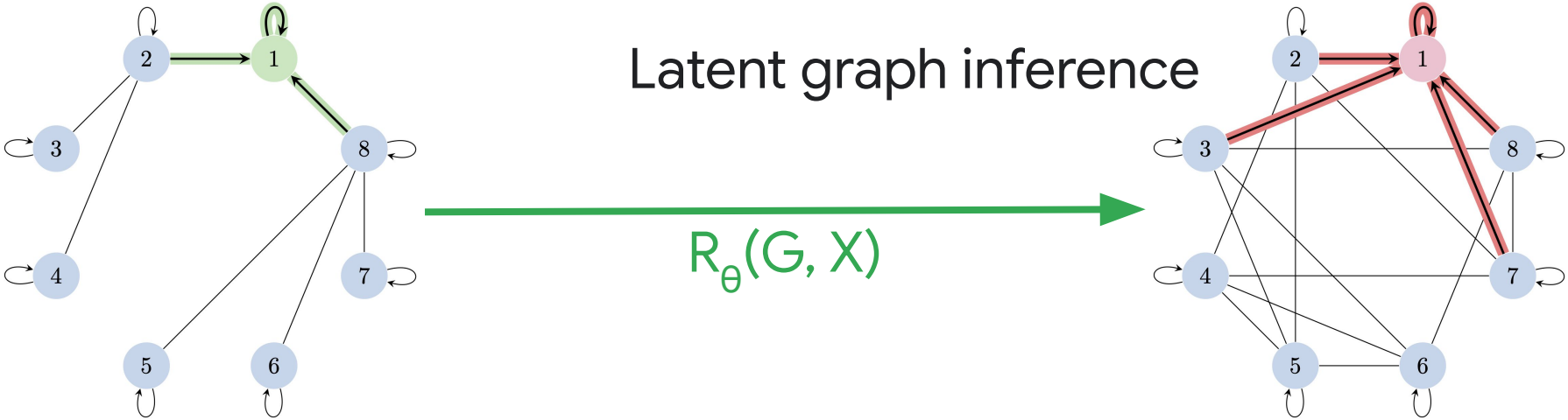
So... how do we *rewire* a graph?

Assume ***no*** edges → Deep Sets

Assume ***all*** edges → Transformers

Nonparametric rewiring

Latent graph inference



So… how do we *rewire* a graph?

Ignores **wealth** of information

Assume ***no*** edges → Deep Sets

Hard to **scale**, hard to **generalise**

Assume ***all*** edges → Transformers

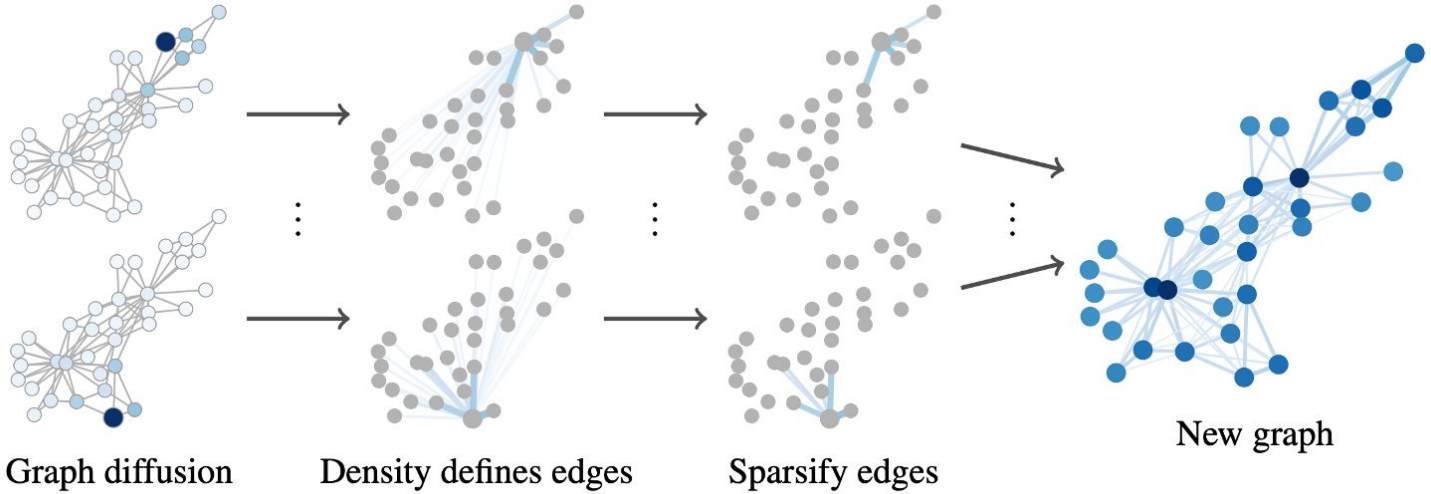
Nonparametric rewiring

Latent graph inference

Discrete decisions – hard to backpropagate!

Option A: *Diffuse* the input graph structure

Graph Diffusion Convolution (**GDC**; Gasteiger *et al.*, NeurIPS'19)



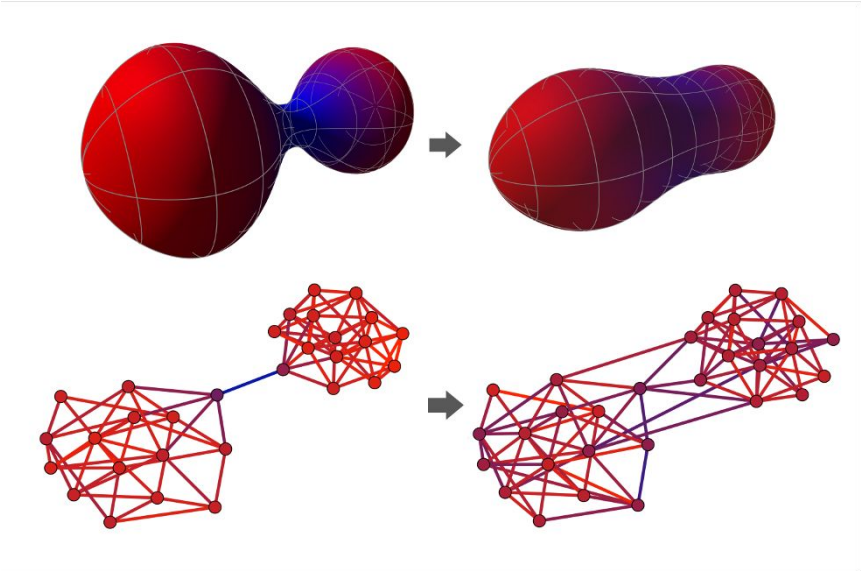
A great place to look for new edges is the input graph itself :)

Use Personalised PageRank diffusion to **diffuse** the edges of the input graph

Elegant and versatile, but relies on ***homophily***, and drastically changes ***statistics***

Option B: *Surgically* rewire the input graph

Stochastic Discrete Ricci Flow (**SDRF**; Topping, Di Giovanni *et al.*, ICLR'22)



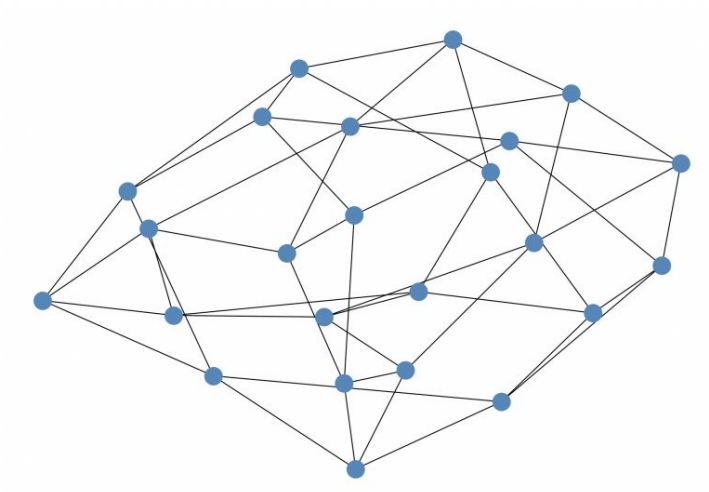
Key observation: edges with **negative curvature** are likely to be bottlenecked

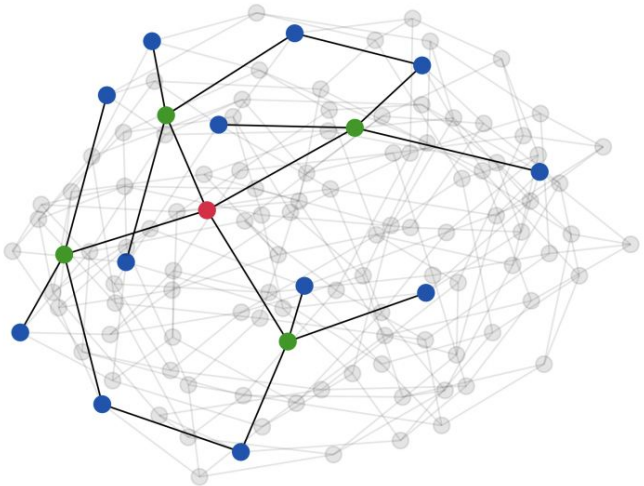
Find edges with most negative Ricci curvature, and **surgically** add edges around them

Preserves input statistics, but curvature measure is ***local***, and ***expensive*** to precompute

Option C: *Precompute* a propagation template

Expander Graph Propagation (**EGP**; Deac *et al.*, LoG'22)





A "brutal" approach: generate a graph which does not need to have **any** relationship to the input graph, **but** one that is excellent at globally-propagating information *(expander graph)*

Alternate GNN layers over the original graph and the expander

(Graph) Transformers a special case — fully-connected graphs are trivial dense expanders

Works well in practice! (***no*** *tuning :))*

Table 3: Statistics of the three graph classification datasets studied in our evaluation.

Name	Number of graphs	Avg. nodes/graph	Avg. edges/graph	Metric
`ogbg-molhiv`	41,127	25.5	27.5	ROC-AUC
`ogbg-molpcba`	437,929	26.0	28.1	Avg. precision
`ogbg-ppa`	158,100	243.4	2,266.1	Accuracy
`ogbg-code2`	452,741	125.2	124.2	F_1 score

Table 2: Comparative evaluation performance on the four datasets studied. Our baseline model is a GIN [56], using exactly the same implementation as in [57]. See Appendix E for ablations.

Model	`ogbg-molhiv`	`ogbg-molpcba`	`ogbg-ppa`	`ogbg-code2`
GIN	0.7558 ± 0.0140	0.2266 ± 0.0028	0.6892 ± 0.0100	0.1495 ± 0.0023

Works well in practice! (***no*** *tuning :))*

Table 3: Statistics of the three graph classification datasets studied in our evaluation.

Name	Number of graphs	Avg. nodes/graph	Avg. edges/graph	Metric
`ogbg-molhiv`	41,127	25.5	27.5	ROC-AUC
`ogbg-molpcba`	437,929	26.0	28.1	Avg. precision
`ogbg-ppa`	158,100	243.4	2,266.1	Accuracy
`ogbg-code2`	452,741	125.2	124.2	F_1 score

Table 2: Comparative evaluation performance on the four datasets studied. Our baseline model is a GIN [56], using exactly the same implementation as in [57]. See Appendix E for ablations.

Model	`ogbg-molhiv`	`ogbg-molpcba`	`ogbg-ppa`	`ogbg-code2`
GIN	0.7558 ± 0.0140	0.2266 ± 0.0028	0.6892 ± 0.0100	0.1495 ± 0.0023
GIN + EGP	$\mathbf{0.7934} \pm 0.0035$	$\mathbf{0.2329} \pm 0.0019$	$\mathbf{0.7027} \pm 0.0159$	$\mathbf{0.1497} \pm 0.0015$

New insights on oversquashing

We are now able to rigorously define over-squashing as the **impedance of mixing** of features between the nodes (Di Giovanni, Rusch, *et al.*, 2023):

Definition 4.1. *Given an MPNN with capacity* (m, w)*, we define the* **over-squashing** *of* v, u *as*

$$\mathrm{OSQ}_{v,u}(m, \mathsf{w}) := \Big(\sum_{k=0}^{m-1} \mathsf{w}^{2m-k-1} \Big(\mathsf{w}(\mathbf{S}^{m-k})^\top \mathrm{diag}(\mathbf{1}^\top \mathbf{S}^k)\mathbf{S}^{m-k} + c^{(2)}\mathbf{Q}_k \Big)_{vu} \Big)^{-1}.$$

Here, *m* is the **depth** of the GNN,

w is the maximal **norm** of its **weights** (e.g. in the message function), and

S/**Q** are functions of the input **graph structure** (adjacency matrix).

This expression depends on all of the below:

- The input graph topology (via **S** and **Q**)
- The GNN architecture (via *m*)
- The GNN parameters (via w)

New insights on oversquashing

We are now able to rigorously define over-squashing as the **impedance of mixing** of features between the nodes (Di Giovanni, Rusch, *et al.,* 2023):

Definition 4.1. *Given an MPNN with capacity* (m, w)*, we define the* **over-squashing** *of* v, u *as*

$$\mathsf{OSQ}_{v,u}(m,\mathsf{w}) := \Big(\sum_{k=0}^{m-1} \mathsf{w}^{2m-k-1} \Big(\mathsf{w}(\mathbf{S}^{m-k})^\top \mathrm{diag}(\mathbf{1}^\top \mathbf{S}^k)\mathbf{S}^{m-k} + c^{(2)}\mathbf{Q}_k \Big)_{vu} \Big)^{-1}.$$

Here, *m* is the **depth** of the GNN,

w is the maximal **norm** of its **weights** (e.g. in the message function), and

S/**Q** are functions of the input **graph structure** (adjacency matrix).

We prove that it is **necessary** for OSQ to be less than the **required mixing** under label y_G

$$\mathsf{OSQ}_{v,u}(m,\mathsf{w}) < (\mathsf{mix}_{y_G}(v,u))^{-1}$$

The OSQ measure: some useful results!

Definition 4.1. *Given an MPNN with capacity* (m, w)*, we define the* **over-squashing** *of* v, u *as*

$$\mathsf{OSQ}_{v,u}(m, \mathsf{w}) := \Big(\sum_{k=0}^{m-1} \mathsf{w}^{2m-k-1} \Big(\mathsf{w}(\mathbf{S}^{m-k})^\top \mathrm{diag}(\mathbf{1}^\top \mathbf{S}^k)\mathbf{S}^{m-k} + c^{(2)}\mathbf{Q}_k \Big)_{vu} \Big)^{-1}.$$

- If we set w = ∞ (infinite capacity in the weights), OSQ = 0
  - *“With large enough weights, I can separate anything”*

The OSQ measure: some useful results!

Definition 4.1. *Given an MPNN with capacity* (m, w)*, we define the* **over-squashing** *of* v, u *as*

$$\mathrm{OSQ}_{v,u}(m, \mathsf{w}) := \left(\sum_{k=0}^{m-1} \mathsf{w}^{2m-k-1} \left(\mathsf{w}(\mathbf{S}^{m-k})^\top \mathrm{diag}(\mathbf{1}^\top \mathbf{S}^k)\mathbf{S}^{m-k} + c^{(2)}\mathbf{Q}_k \right)_{vu} \right)^{-1}.$$

- If we set w = ∞ (infinite capacity in the weights), OSQ = 0
  - *“With large enough weights, I can separate anything”*
- If we set w = 0 (zero weights), OSQ = ∞
  - *“Without weights, every message is a constant, and hence cannot usefully mix”*

The OSQ measure: some useful results!

Definition 4.1. *Given an MPNN with capacity* (m, w)*, we define the* **over-squashing** *of* v, u *as*

$$\mathrm{OSQ}_{v,u}(m, \mathsf{w}) := \Big(\sum_{k=0}^{m-1} \mathsf{w}^{2m-k-1} \Big(\mathsf{w}(\mathbf{S}^{m-k})^\top \mathrm{diag}(\mathbf{1}^\top \mathbf{S}^k)\mathbf{S}^{m-k} + c^{(2)}\mathbf{Q}_k \Big)_{vu} \Big)^{-1}.$$

- If we set w = ∞ (infinite capacity in the weights), OSQ = 0
  - *“With large enough weights, I can separate anything”*
- If we set w = 0 (zero weights), OSQ = ∞
  - *“Without weights, every message is a constant, and hence cannot usefully mix”*
- If we set $2m < d(u, v)$ (under-reaching), OSQ = ∞
  - *“I cannot mix two nodes together if the GNN doesn’t reach one from the other”*

The OSQ measure: some useful results!

Definition 4.1. *Given an MPNN with capacity* (m, w)*, we define the* **over-squashing** *of* v, u *as*

$$\mathsf{OSQ}_{v,u}(m,\mathsf{w}) := \Big(\sum_{k=0}^{m-1} \mathsf{w}^{2m-k-1} \Big(\mathsf{w}(\mathbf{S}^{m-k})^\top \mathrm{diag}(\mathbf{1}^\top \mathbf{S}^k)\mathbf{S}^{m-k} + c^{(2)}\mathbf{Q}_k \Big)_{vu} \Big)^{-1}.$$

- If we set w = ∞ (infinite capacity in the weights), OSQ = 0
  - *“With large enough weights, I can separate anything”*
- If we set w = 0 (zero weights), OSQ = ∞
  - *“Without weights, every message is a constant, and hence cannot usefully mix”*
- If we set 2*m* < d(*u*, *v*) (under-reaching), OSQ = ∞
  - *“I cannot mix two nodes together if the GNN doesn’t reach one from the other”*
- If we fix *m* and let w vary, weights need to be large enough to allow mixing:

Theorem 4.2. *Let* $\mathbf{A} = \mathbf{A}_{\mathrm{sym}}$*,* $r := d_{\mathsf{G}}(v,u)$*,* $m = \lceil r/2 \rceil$*, and* q *be the number of paths of length* r *between* v *and* u*. For an MPNN satisfying Theorem 3.2 with capacity* $(m = \lceil r/2 \rceil, \mathsf{w})$*, we find* $\mathsf{OSQ}_{v,u}(m,\mathsf{w}) \cdot (c_2\mathsf{w})^r (\mathbf{A}^r)_{vu} \geq 1$*. In particular, if the MPNN generates mixing* $\mathsf{mix}_{y_{\mathsf{G}}}(v,u)$*, then*

$$\mathsf{w} \geq \frac{d_{\min}}{c_2} \left(\frac{\mathsf{mix}_{y_{\mathsf{G}}}(v,u)}{q} \right)^{\frac{1}{r}}.$$

The OSQ measure: some useful results!

Definition 4.1. *Given an MPNN with capacity* (m, w)*, we define the* **over-squashing** *of* v, u *as*

$$\mathrm{OSQ}_{v,u}(m,\mathsf{w}) := \Big(\sum_{k=0}^{m-1} \mathsf{w}^{2m-k-1} \Big(\mathsf{w}(\mathbf{S}^{m-k})^\top \mathrm{diag}(\mathbf{1}^\top \mathbf{S}^k)\mathbf{S}^{m-k} + c^{(2)}\mathbf{Q}_k \Big)_{vu} \Big)^{-1}.$$

- If we set w = ∞ (infinite capacity in the weights), OSQ = 0
  - *“With large enough weights, I can separate anything”*
- If we set w = 0 (zero weights), OSQ = ∞
  - *“Without weights, every message is a constant, and hence cannot usefully mix”*
- If we set 2*m* < d(*u*, *v*) (under-reaching), OSQ = ∞
  - *“I cannot mix two nodes together if the GNN doesn’t reach one from the other”*
- If we fix *m* and let w vary, weights need to be large enough to allow mixing:

d-ary **Tree** of depth *r*; *m* = *r*/2

$$\mathsf{w} \geq (d+1)\left(\frac{y(v,u)}{d+1}\right)^{\frac{1}{r}}.$$

Grows with *d*! Graphs are *harder* than **paths** :)

Complete graph of *n* nodes; *m* = 1

$$\mathsf{w} \geq (n-1)y(v,u).$$

Grows with *n*! Redundant messages are a problem.

The OSQ measure: some useful results!

Definition 4.1. *Given an MPNN with capacity* (m, w)*, we define the* **over-squashing** *of* v, u *as*

$$\mathsf{OSQ}_{v,u}(m,\mathsf{w}) := \Big(\sum_{k=0}^{m-1} \mathsf{w}^{2m-k-1} \Big(\mathsf{w}(\mathbf{S}^{m-k})^\top \mathrm{diag}(\mathbf{1}^\top \mathbf{S}^k)\mathbf{S}^{m-k} + c^{(2)}\mathbf{Q}_k \Big)_{vu} \Big)^{-1}.$$

- If we set w = ∞ (infinite capacity in the weights), OSQ = 0
  - *“With large enough weights, I can separate anything”*
- If we set w = 0 (zero weights), OSQ = ∞
  - *“Without weights, every message is a constant, and hence cannot usefully mix”*
- If we set 2*m* < d(*u*, *v*) (under-reaching), OSQ = ∞
  - *“I cannot mix two nodes together if the GNN doesn’t reach one from the other”*
- If we fix *m* and let w vary, weights need to be large enough to allow mixing
  - *But too large w is numerically unstable!*
- If we fix w and let *m* vary, GNN needs to be deep enough to allow mixing

Theorem 4.3. *Consider an MPNN satisfying Theorem 3.2, with* $\max\{\mathsf{w}, \omega/\mathsf{w} + c_1\gamma + c_2\} \leq 1$*, and* $\mathbf{A} = \mathbf{A}_{\mathrm{sym}}$*. If* $\mathsf{OSQ}_{v,u}(m,\mathsf{w}) \cdot (\mathsf{mix}_{y_\mathsf{G}}(v,u)) \leq 1$*, and hence the MPNN generates mixing* $\mathsf{mix}_{y_\mathsf{G}}(v,u)$ *among the features associated with nodes* v, u*, then the number of layers* m *satisfies*

$$m \geq \frac{\tau(v,u)}{4c_2} + \frac{|\mathsf{E}|}{\sqrt{d_v d_u}} \Big(\frac{\mathsf{mix}_{y_\mathsf{G}}(v,u)}{\gamma\mu} - \frac{1}{c_2} \Big(\frac{\gamma + |1 - c_2\lambda^*|^{r-1}}{\lambda_1} + 2\frac{c^{(2)}}{\mu} \Big) \Big),$$

The OSQ measure: some useful results!

Definition 4.1. *Given an MPNN with capacity* (m, w)*, we define the* **over-squashing** *of* v, u *as*

$$\mathrm{OSQ}_{v,u}(m, \mathsf{w}) := \Big(\sum_{k=0}^{m-1} \mathsf{w}^{2m-k-1} \Big(\mathsf{w}(\mathbf{S}^{m-k})^\top \mathrm{diag}(\mathbf{1}^\top \mathbf{S}^k)\mathbf{S}^{m-k} + c^{(2)}\mathbf{Q}_k \Big)_{vu} \Big)^{-1}.$$

- If we set w = ∞ (infinite capacity in the weights), OSQ = 0
  - *"With large enough weights, I can separate anything"*
- If we set w = 0 (zero weights), OSQ = ∞
  - *"Without weights, every message is a constant, and hence cannot usefully mix"*
- If we set $2m < \mathrm{d}(u, v)$ (under-reaching), OSQ = ∞
  - *"I cannot mix two nodes together if the GNN doesn't reach one from the other"*
- If we fix *m* and let w vary, weights need to be large enough to allow mixing
  - *But too large w is numerically unstable!*
- If we fix w and let *m* vary, GNN needs to be deep enough to allow mixing
  - *This depth must be **at least** on the order of the* **commute time** $\tau(v, u)$
  - *This is often **significantly** larger than the shortest-path distance – sometimes* $O(n^3)$*!*

$$m \geq \boxed{\frac{\tau(v,u)}{4c_2}} + \frac{|\mathsf{E}|}{\sqrt{d_v d_u}} \Big(\frac{\mathsf{mix}_{y_\mathsf{G}}(v,u)}{\gamma\mu} - \frac{1}{c_2} \Big(\frac{\gamma + |1 - c_2\lambda^*|^{r-1}}{\lambda_1} + 2\frac{c^{(2)}}{\mu} \Big) \Big),$$

The OSQ measure: some useful results!

Definition 4.1. *Given an MPNN with capacity* (m, w)*, we define the* **over-squashing** *of* v, u *as*

$$\mathrm{OSQ}_{v,u}(m, \mathsf{w}) := \Big(\sum_{k=0}^{m-1} \mathsf{w}^{2m-k-1} \Big(\mathsf{w}(\mathbf{S}^{m-k})^\top \mathrm{diag}(\mathbf{1}^\top \mathbf{S}^k)\mathbf{S}^{m-k} + c^{(2)}\mathbf{Q}_k \Big)_{vu} \Big)^{-1}.$$

- If we set w = ∞ (infinite capacity in the weights), OSQ = 0
  - *"With large enough weights, I can separate anything"*
- If we set w = 0 (zero weights), OSQ = ∞
  - *"Without weights, every message is a constant, and hence cannot usefully mix"*
- If we set $2m < \mathrm{d}(u, v)$ (under-reaching), OSQ = ∞
  - *"I cannot mix two nodes together if the GNN doesn't reach one from the other"*
- If we fix *m* and let w vary, weights need to be large enough to allow mixing
  - *But too large w is numerically unstable!*
- If we fix w and let *m* vary, GNN needs to be deep enough to allow mixing
  - *This depth must be* ***at least*** *on the order of the* ***commute time*** $\tau(v, u)$
  - *This is often* ***significantly*** *larger than the shortest-path distance – sometimes* $O(n^3)$*!*
  - ***Expander graphs*** *have very favourable commute times—justifying EGP :)*

$$m \geq \boxed{\frac{\tau(v,u)}{4c_2}} + \frac{|\mathsf{E}|}{\sqrt{d_v d_u}} \Big(\frac{\mathsf{mix}_{y_\mathsf{G}}(v,u)}{\gamma\mu} - \frac{1}{c_2} \Big(\frac{\gamma + |1 - c_2\lambda^*|^{r-1}}{\lambda_1} + 2\frac{c^{(2)}}{\mu} \Big) \Big),$$

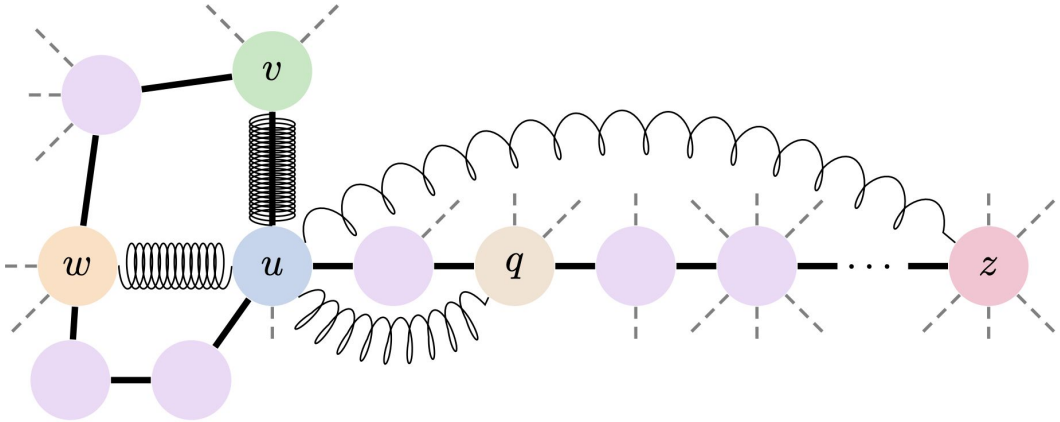
In conclusion...

- In most cases, we don't have **fully correct** graphs to give our GNNs
  - And even when we do, we'd likely benefit from **specialising** them to the task at hand

In conclusion...

- In most cases, we don't have **fully correct** graphs to give our GNNs
  - And even when we do, we'd likely benefit from **specialising** them to the task at hand

- Not all graphs are made equal, in terms of their **suitability** for message passing
  - Graphs with *high commute times* are **detrimental** to mixing, no matter the GNN*!



*our analysis does not yet cover GNNs with non-sum aggregators

In conclusion...

- In most cases, we don't have **fully correct** graphs to give our GNNs
  - And even when we do, we'd likely benefit from **specialising** them to the task at hand

- Not all graphs are made equal, in terms of their **suitability** for message passing
  - Graphs with *high commute times* are **detrimental** to mixing, no matter the GNN*!

- Rewiring likely to be a sufficient "missing piece" to explaining ***any*** discrete DL architecture
  - Hence, choosing the computational graph is a **key** unsolved problem in deep learning

*our analysis does not yet cover GNNs with non-sum aggregators

In conclusion...

- In most cases, we don't have **fully correct** graphs to give our GNNs
  - And even when we do, we'd likely benefit from **specialising** them to the task at hand

- Not all graphs are made equal, in terms of their **suitability** for message passing
  - Graphs with *high commute times* are **detrimental** to mixing, no matter the GNN*!

- Rewiring likely to be a sufficient "missing piece" to explaining ***any*** discrete DL architecture
  - Hence, choosing the computational graph is a **key** unsolved problem in deep learning

- Currently, **nonparametric** rewiring hits the "sweet spot" between *tractability* and *elegance*
  - Can ***diffuse*** the graph, surgically ***rewire*** it, or use a known good template (***expander***)

*our analysis does not yet cover GNNs with non-sum aggregators

In conclusion...

- In most cases, we don't have **fully correct** graphs to give our GNNs
  - And even when we do, we'd likely benefit from **specialising** them to the task at hand
- Not all graphs are made equal, in terms of their **suitability** for message passing
  - Graphs with *high commute times* are **detrimental** to mixing, no matter the GNN*!
- Rewiring likely to be a sufficient "missing piece" to explaining ***any*** discrete DL architecture
  - Hence, choosing the computational graph is a **key** unsolved problem in deep learning
- Currently, **nonparametric** rewiring hits the "sweet spot" between *tractability* and *elegance*
  - Can ***diffuse*** the graph, surgically ***rewire*** it, or use a known good template (***expander***)
- We have a *formula* for **over-squashing**, which you can use to assess suitability of graphs

*our analysis does not yet cover GNNs with non-sum aggregators

Thank you.

Petar Veličković
petarv@google.com
https://petar-v.com

Google DeepMind

With many thanks to Michael Bronstein, Andreea Deac, Francesco Di Giovanni, Marc Lackenby, Siddhartha Mishra and Konstantin Rusch